Algebraic Operations on PQ Trees and Modular Decomposition Trees

Ross M. McConnell [*] Fabien de Montgolfier [†]

Abstract

Partitive set families are families of sets that can be quite large, but have a compact, recursive representation in the form of a tree. This tree is a common generalization of PQ trees, the modular decomposition of graphs, certain decompositions of boolean functions, and decompositions that arise on a variety of other combinatorial structures. We describe natural operators on partitive set families, give algebraic identities for manipulating them, and describe efficient algorithms for evaluating them. We use these results to obtain new time bounds for finding the common intervals of a set of permutations, finding the modular decomposition of an edge-colored graph (also known as a two-structure), finding the PQ tree of a matrix when a consecutive-ones arrangement is given, and finding the modular decomposition of a permutation graph when its permutation realizer is given.

1 Introduction

A 0-1 matrix has the *consecutive-ones property* if there exists a permutation of the set of columns such that the 1's in each row occupy a consecutive block. Such a permutation is called a *consecutive-ones ordering* (Figure 1).

In general, the number of consecutive-ones orderings need not be polynomial; there may be $|V|!$ of them. However, the *PQ tree* of a family that has the consecutive-ones property gives a way to represent all of its consecutive-ones orderings using $O(|V|)$ space, as in Figure 1. The PQ tree is a rooted, ordered tree whose leaves are the elements of V, and whose internal nodes are each labeled either P or Q. The left-to-right leaf order gives a consecutive-ones ordering, and any new leaf order that can be obtained by permuting arbitrarily the children of a P node or reversing the order of children of a Q node is also a consecutive-ones ordering. There are no other consecutive-ones orderings.

One of the most significant applications of PQ trees is in finding planar embeddings of planar graphs. Booth and Lueker used PQ trees to develop an algorithm for determining whether a family of sets has the consecutive-ones property [2]. The algorithm runs in $O(|V|+l(\mathcal{F}))$ time, where $l(\mathcal{F})$ is the sum of cardinalities of members of $\mathcal{F}$, or *length* of $\mathcal{F}$.

A set family $\mathcal{F}$ with the consecutive-ones property gives

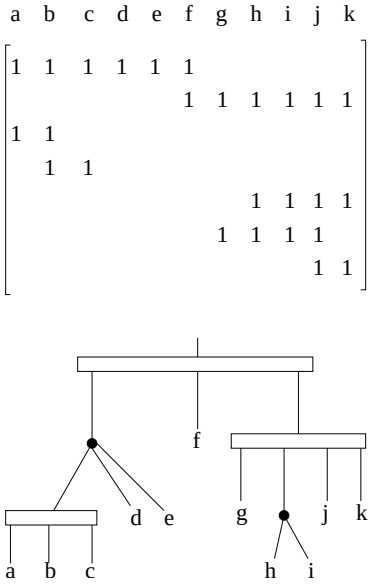

Figure 1: A consecutive-ones ordering of a matrix, and the corresponding PQ tree. The zeros in the matrix are omitted. The ordering of the columns is a consecutive-ones ordering because the 1's in each row are consecutive. The left-to-right leaf order of the PQ tree gives this ordering. Reversing the left-to-right order of children of a Q node (rectangles) or permuting arbitrarily the left-to-right order of children of a P node (points) induces a new leaf order, which is also a consecutive-ones ordering. For instance, permuting the order of children of the left child of the root and reversing the order of children of the right child gives (d,a,b,c,e,f,k,j,h,i,g) as a consecutive-ones ordering. An ordering of columns of the matrix is a consecutive-ones ordering iff it is the leaf order of the PQ tree induced by reversing the children of some set of Q nodes and permuting the children of some set of P nodes.

[*]Colorado State University rmm@cs.colostate.edu
[†]Université Paris 7 mf@liafa.jussieu.fr

rise to an *interval graph*, which has one vertex for each member of $\mathcal{F}$, and an adjacency between two vertices if and only if the corresponding members of $\mathcal{F}$ intersect. Booth and Lueker's result gave a linear-time algorithm for determining whether a given graph is an interval graph, and, if so, finding such a set family $\mathcal{F}$ for it. This problem played a key role during the 1950's in establishing that DNA has a linear topology [1], though linear-time algorithms were unavailable at that time. Variations on this problem come up in the physical mapping of a genome, using laboratory data that can be modeled with a graph [20, 25].

A *module* of an undirected graph $G = (V, E)$ is a set X of vertices such that each vertex $y \in V - X$ is either adjacent to all members of X or adjacent to none of them. The number of modules can be exponential in the size of G. However, there exists a compact $O(|V|)$ representation of all the modules, called the *modular decomposition*. The modular decomposition, first described by Gallai [11], is a tree that has the members of V as its leaves, and where the internal nodes are all labeled *prime* or *degenerate*. Details of the representation are given below. The modular decomposition can be computed in $O(|V| + |E|)$ time [17].

A close relationship between the modular decomposition and a variety of combinatorial problems on graphs have been described. Gallai [11] showed a close relationship to the *transitive orientation problem*, which is the problem of orienting the edges of an undirected edge so that the resulting digraph is transitive (*i.e.* a poset relation). Using the modular decomposition, a transitive orientation, if it exists, can be found in $O(|V| + |E|)$ time [17]. This result has led to linear time bounds for maximum clique and minimum coloring on transitively orientable graphs (*i.e.* comparability graphs), and recognizing permutation graphs and co-interval graphs. Surveys on applications can be found in [21, 22, 23].

The modular decomposition has a straightforward extension to directed graphs, and linear time bounds have recently been given for finding it [18].

The modules of a graph are an example of a *partitive set family* [4, 21]. All partitive set families have a compact representation by means of a tree; the modular decomposition is just an example of it when the set family is the modules of a graph. The PQ tree is another example of this phenomenon. In [16], it is shown that the PQ tree is this representation of a certain partitive family defined by the 0-1 matrix, and, more generally that, like the modular decomposition, the PQ tree is an example of a *substitution decomposition* [22], a combinatorial abstraction that has partitive families as a central ingredient.

Other partitive families have played a role in linear time bounds for recognizing circular-arc graphs [15, **?**], $O(n + m \log n)$ bounds for recognizing probe interval graphs [19], and arise in decompositions of boolean expressions [22].

In this paper, we describe natural algebraic operators on decomposition trees of partitive families, give identities for manipulating them, and develop algorithms for evaluating them. We use these results to obtain new time bounds for combinatorial problems that involve partitive families, such as finding the *common intervals* of a set of permutations [**?**, **?**], finding the modular decomposition of edge-colored graphs, or *two-structures* [9], finding the PQ tree of a matrix when a consecutive-ones arrangement is given, and finding the modular decomposition of a permutation graph when its realizer in the form of a permutation is given.

2 Preliminaries

Two sets X and Y *overlap* if they intersect, but neither is a subset of the other. That is, they overlap if $X - Y$, $Y - X$, and $X \cap Y$ are all nonempty.

Let $\mathcal{F}$ is a family of subsets of a set V. Then let $|\mathcal{F}|$ denote the number of sets in $\mathcal{F}$; this contrasts with $l(\mathcal{F})$, which is the sum of cardinalities of the sets in $\mathcal{F}$.

In general, it takes $\Omega(l(\mathcal{F}))$ space to represent $\mathcal{F}$ in the computer. However, suppose that $\mathcal{F}$ satisfies the following: $V \in \mathcal{F}$, $\{x\} \in \mathcal{F}$ for all $x \in V$, and no two members of $\mathcal{F}$ overlap. In this case, it is easy to see that $l(\mathcal{F})$ can be $\Omega(|V|^2)$, but $\mathcal{F}$ can be represented in $O(|V|)$ space. The Hasse diagram of the subset relation on members of $\mathcal{F}$ is a tree whose root is V and whose leaves are its one-element subsets. Labeling only the leaves of this tree with the corresponding set gives a representation of $\mathcal{F}$. Given a node of the tree, the set X that it represents can be returned in $O(|X|)$ time by traversing its subtree and assembling the disjoint union of its leaf descendants. This is as efficient as any representation of X, but takes $O(1)$ space to represent X.

Let us call such a set family a *tree-like* family, and its tree representation its *inclusion tree*. Partitive families are a generalization of tree-like families, called *partitive families*, that may have a number of members that is exponential in the size of V, yet still has an $O(|V|)$ representation.

DEFINITION 2.1. *[11, 4, 22, 9] A set family $\mathcal{F}$ on domain V is partitive iff it has the following properties:*

- $V \in \mathcal{F}$, $\emptyset \notin \mathcal{F}$, *and for all* $v \in V$, $\{v\} \in \mathcal{F}$
- *For all* $X, Y \in \mathcal{F}$, *if* X *and* Y *overlap, then* $X \cap Y \in \mathcal{F}$, $X \cup Y \in \mathcal{F}$, $X - Y \in \mathcal{F}$, *and* $Y - X \in \mathcal{F}$.

Let the *strong members* of a partitive family be those that overlap with no other member of $\mathcal{F}$, and let the *weak members* be the remaining members.

THEOREM 2.1. *[4, 22] The strong members of a partitive family $\mathcal{F}$ are a tree-like family where the Hasse diagram T of the subset relation has the following properties:*

1. *Every weak member of $\mathcal{F}$ is a union of siblings in T;*

2. *Each internal node* X *can be classified as one of the following types:*

   (a) *Degenerate: Every union of more than one child is a member of* $\mathcal{F}$ *;*

   (b) *Prime: Other than* X *itself, no union of more than one child is a member of* $\mathcal{F}$ *;*

   (c) *Linear: There exists a linear order on the children such that a union of more than one child is a member of* $\mathcal{F}$ *if and only if the children are consecutive in the linear order.*

Conversely, every set family that has such a representative is partitive. Let us call the tree representation of $\mathcal{F}$ given by the theorem the *decomposition tree* of $\mathcal{F}$.

EXAMPLE 2.1. *Removing the empty set from the power set of* V *gives a partitive family. Its decomposition tree has one internal node,* V*, which is degenerate, and one leaf for each member of* V*.*

EXAMPLE 2.2. *A nonempty set* X *of vertices of a directed graph* $G = (V, E)$ *is a* module *iff it satisfies the following conditions for to every* $y \in V - X$*:*

1. *Either every element of* X *is a neighbor of* y *or no element of* X *is a neighbor of* y*;*

2. *Either* y *is a neighbor of every element of* X *or a neighbor of no element of* X*.*

It is not hard to show that the modules of a graph satisfy the requirements of Definition 2.1. It follows that the modules of a graph can be represented in $O(|V|)$ *space with a tree [11, 22]. As illustrated by the complete graph on* V*, the number of modules can be exponential in* V*. Figure 2 gives a nontrivial example. This tree is the* modular decomposition *of the graph. It takes* $O(|V| + |E|)$ *time to compute the modular decomposition of an arbitrary directed graph [18]; linear time bounds for the special case of undirected graphs were given in [17].*

If $\mathcal{F}$ is a partitive set family, let $T(\mathcal{F})$ denote its decomposition tree, and if T is a partitive decomposition tree, let $\mathcal{F}(T)$ denote the set family that it represents. There is no way to distinguish whether a node with two children is prime, degenerate or linear, but the classification is unique for nodes with three or more children. Henceforth, we will consider a node to be classified as prime, degenerate, or linear only if it has at least three children.

DEFINITION 2.2. *A partitive set family is* symmetric *if, whenever* X *and* Y *are overlapping members of* $\mathcal{F}$*, the* symmetric difference $X \Delta Y = (X - Y) \cup (Y - X)$ *is a member of* $\mathcal{F}$*. It is* antisymmetric *if* $X \Delta Y$ *is never a member when* X *and* Y *are overlapping members.*

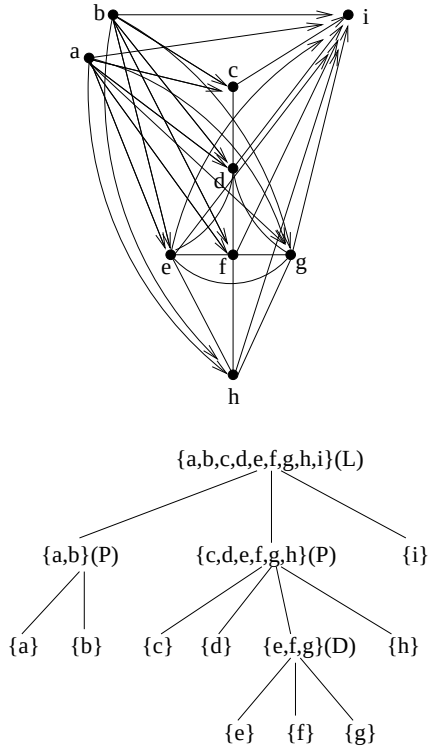

Figure 2: The modular decomposition represents the modules of a graph with an ordered tree whose nodes are subsets of V. Each internal node is labeled *linear* (L), *prime* (P), or *degenerate*(D). A subset of the vertices is a module iff it is a node of the tree, the union of a set of children of a degenerate node, or the union of a set of children of a linear node that is consecutive in the left-to-right order of its children. The modules of the depicted graph that are not nodes of the tree are unions of children of $\{e, f, g\}$, namely, $\{e, f\}$, $\{e, g\}$, and $\{f, g\}$, and unions of consecutive children of $\{a, b, c, d, e, f, g, h, i\}$, namely, $\{a, b, c, d, e, f, g, h\}$ and $\{c, d, e, f, g, h, i\}$. To represent the decomposition, it is not necessary to label internal nodes with the set that they correspond to, as this is given by the union of leaf descendants of the node.

.

It is not hard to see that if a graph is symmetric (undirected), its modules are a symmetric partitive family, and that if it is antisymmetric, its modules are an antisymmetric partitive family, unless it has modules that induce disconnected subgraphs. In particular, the modules of a tournament are an antisymmetric partitive family. A partitive set family is symmetric if and only if its decomposition tree has no linear nodes, and it is antisymmetric if and only if its decomposition tree has no degenerate nodes.

If $\mathcal{F}$ is an arbitrary set family, let $\mathcal{C}(\mathcal{F})$ denote the *partitive closure* of $\mathcal{F}$, namely, the smallest partitive family $\mathcal{F}'$ such that $\mathcal{F} \subseteq \mathcal{F}'$. Let $\mathcal{S}(\mathcal{F})$ denote the *symmetric partitive closure* of $\mathcal{F}$, namely, the smallest symmetric partitive family $\mathcal{F}''$ such that $\mathcal{F} \subseteq \mathcal{F}''$. It is shown in [16] that each of these closures is unique.

If M is a zero-one matrix, then let V denote its columns, and let $\mathcal{F}(M)$ denote the set family on V that has one set for each row of M, namely, the one obtained by interpreting the row as the bit-vector representation of a set. That is, the set represented by a row is the set of columns where the row has a 1. Conversely, if $\mathcal{F}$ is a family of subsets of a domain V, we may obtain a representation of $\mathcal{F}$ with M, such that $\mathcal{F}(M) = \mathcal{F}$. M has the consecutive-ones property if and only if there exists an ordering of V such that every member of $\mathcal{F}$ is consecutive, and, in this case, we may refer to the PQ tree of M as the the PQ tree of $\mathcal{F}$.

The following gives a generalization of the PQ tree to arbitrary set families or zero-one matrices:

DEFINITION 2.3. *[16] Let $\mathcal{F}$ be an arbitrary set family. Let the* PQR tree *of $\mathcal{F}$ be the decomposition tree of $\mathcal{C}(\mathcal{F})$, where the prime nodes are labeled P, the linear nodes are labeled Q, and the degenerate nodes are labeled R.*

THEOREM 2.2. *[16] $\mathcal{F}$ has the consecutive-ones property if and only if its PQR tree has no R nodes, and, in this case, its PQR tree is its PQ tree.*

Let $\mathcal{F}$ be an arbitrary set family on V, and $\mathcal{N}(\mathcal{F})$ denote the family of nonempty subsets of V that don't overlap with any member of $\mathcal{F}$.

THEOREM 2.3. *[14] $\mathcal{N}(\mathcal{F})$ is a symmetric partitive set family, and if $\mathcal{F}$ has the consecutive-ones property, its decomposition tree is the PQ tree, where the prime nodes are interpreted as Q nodes and the degenerate nodes are interpreted as P nodes.*

The proof of the following is elementary:

THEOREM 2.4. *[18] If $\mathcal{F}_1$ and $\mathcal{F}_2$ are two partitive families, then so is $\mathcal{F}_1 \cap \mathcal{F}_2$. If they are both symmetric partitive families, then so is $\mathcal{F}_1 \cap \mathcal{F}_2$, and if they are both antisymmetric partitive families, then so if $\mathcal{F}_1 \cap \mathcal{F}_2$.*

DEFINITION 2.4. *If T_1 and T_2 are partitive decomposition trees, then let $T_1 \cap T_2$ be the decomposition tree of $\mathcal{F}(T_1) \cap \mathcal{F}(T_2)$, which exists by Theorem 2.4.*

THEOREM 2.5. *[18] Given decomposition trees T_1 and T_2 of symmetric partitive families, it takes time proportional to the sum of cardinalities of their nodes to find $T_1 \cap T_2$.*

3 New Results

3.1 Intersection of Arbitrary Partitive Families. Theorem 2.5 applies only to symmetric partitive families. The case where they are not symmetric is more difficult.

The additional difficulties posed by linear nodes are illustrated by the simple case of two trees T_1 and T_2 that each have $V = \{1,2,..,8\}$ as their only internal node. If T_1 and T_2 are decomposition trees of symmetric partitive families, then V is prime or degenerate in each. In $T_1 \cap T_2$, V is the only internal node, and it is degenerate if it is degenerate in both trees and prime otherwise. The intersection is trivial to compute in this case.

On the other hand, suppose V is linear in each of T_1 and T_2, and $(\{1\},\{2\},...,\{8\})$ is the order of its children in T_1 and $(\{6\},\{5\},\{8\},\{7\},\{2\},\{1\},\{4\},\{3\})$ is the order of children in T_2. Then $\{1,2\}$, $\{3,4\}$, $\{5,6\}$, $\{7,8\}$, $\{1,2,3,4\}$, and $\{5,6,7,8\}$ are internal nodes. In general, two linear nodes in two partitive trees can give rise to a complicated subtree in the intersection.

We improve the bound of Theorem 2.4 and generalize it to arbitrary partitive families, not just symmetric ones:

THEOREM 3.1. *Given arbitrary partitive decomposition trees T_1 and T_2 on domain V, it takes $O(|V|)$ time to find $T_1 \cap T_2$.*

Let the *intervals* of a permutation $(\nu_1,\nu_2,...,\nu_n)$ be a nonempty set of the form $\{\nu_i,\nu_{i+1},...,\nu_j\}$. The *common intervals* of a set of permutations of the same set are the intervals that are common to all of them.

If π is a linear ordering of V, then its intervals are an antisymmetric partitive family: their decomposition tree $T(\pi)$ is the tree with one internal linear node, and leaf set V ordered in the order given by π. It follows that the common intervals of a set $\{\pi_1,\pi_2,...,\pi_k\}$ is a partitive set family whose decomposition tree is given by $T(\pi_1) \cap T(\pi_2) \cap ... \cap T(\pi_k)$.

APPLICATION 3.1. *The following is immediate from this example and Theorem 3.1:*

THEOREM 3.2. *It takes $O(kn)$ time to find the common intervals of a set $\{\pi_1,\pi_2,...,\pi_k\}$ of permutations of a set V.*

The previous time bound for this problem was $O(nk+K)$, where K is the number of common intervals [**?**]. (Note that K can be quadratic in n).

APPLICATION 3.2. *The conceptual complexity of many linear-time algorithms for computing the PQ tree is well-known. However, Theorem 3.1 gives a simple* $O(nm)$ *approach to finding the PQ tree of an* $n \times m$ *matrix. Let* M *be a 0-1 matrix with* m *columns and* n *rows, let* M_1 *be the submatrix given by the top* $\lfloor m/2 \rfloor$ *rows and let* M_2 *be the submatrix given by the remaining* $\lceil m/2 \rceil$ *rows. To find the PQ tree of* M*, we may find the PQ trees* T_1 *and* T_2 *of* M_1 *and* M_2 *by recursion, and then return* $T_1 \cap T_2$ *as the PQ tree of* M*. The correctness follows from Theorem 2.3.*

APPLICATION 3.3. *[?] If* T *is a PQ tree, let* $\Pi(T)$ *denote the set of permutations represented by* T*. If* T *and* T' *are PQ trees on domain* V*, then let* $T \preceq T'$ *denote that* $\Pi(T) \subseteq \Pi(T')$*. Given the PQ trees* T_1 *and* T_2 *of two set families on the same domain* V*, let the the* join *of* T_1 *and* T_2 *be the minimal PQ tree* T_3 *(with respect to* $\preceq$*) such that* $T_1 \preceq T_3$ *and* $T_2 \preceq T_3$*.*

The join of two PQ trees was first described by Landau, Parida, and Weimann [?], who have used it in an application to genomics, and who obtained an $O(|V|^3)$ *algorithm to compute it. When we communicated Theorem 3.2, they used it to improve the bound to* $O(|V|)$*.*

A *two-structure* is a directed graph whose edges are colored. A module of a two-structure is a module X of the underlying graph that satisfies the following additional requirement: whenever $y \in V - X$, all edges from members of X to y are the same color, and all edges from y to members of X are the same color. The modules of a two-structure are a partitive family [9].

The following give a relationship between modular decomposition of two-structures and problems in other areas that have not been observed before.

EXAMPLE 3.1. *A distance function* d *on a set* V *is an* ultrametric *if, for all* $x, y, z \in V$*, either* $d(x,y)$*,* $d(y,z)$ *and* $d(x,z)$ *are all equal, or two are equal and the third is smaller. Ultrametrics arise in many clustering applications, such as the problem of inferring phylogenetic trees. An example is the distance metric in a graph with edge weights, where the* height *of a path is the maximum weight of an edge on the path, and where the* distance *between two vertices is the height of the minimum-height path between them. An ultrametric can be modeled as a two-structure, where for* $x, y \in V$*, the "color" of edge* xy *is* $d(x,y)$*. In this case, the modular decomposition of the two structure is the tree returned by the well-known UPGMA clustering algorithm [?].*

EXAMPLE 3.2. *Given a string* $a_1 a_2 ..., a_n$*, let us define a two-structure with vertices* $\{1, 2, ..., n\}$*, and for vertices* i *and* j*, let the label ("color") of edge* ij *be the longest common prefix of* $a_i a_{i+1} ... a_n$ *and* $a_j a_{j+1} ... a_n$*. It is not hard to show that the modular decomposition of this two-structure is the well-known* suffix tree *of the string, which is used in efficient solutions to a variety of combinatorial problems on strings [6, 13].*

Though these last two examples yield an interesting structural relationship, they do not yield more efficient algorithms. However, in [18], we give a linear-time algorithm for finding the modular decomposition of a symmetric (undirected) two-structure. This is a key step in the linear time bounds we show there for finding the modular decomposition of a directed graph.

Because of the added difficulties posed by linear nodes in the decomposition, the best bound until now for finding the modular decomposition of arbitrary two-structures has been $O(|V|^2)$ [8]. However, given Theorem 3.1, we can now improve this quite easily:

PROPOSITION 3.1. *It takes* $O(k|V| + |E|)$ *time to find the modular decomposition of a two-structure that has vertex set* V*, edge set* E*, and* k *edge colors.*

Proof. Let G_i denote the graph on V given by edges of color i. Find the modular decomposition T_i of each G_i for each i from 1 to k using the linear-time modular decomposition algorithm for directed graphs given in [18]. Since the edge sets are disjoint, this takes a total of $O(k|V| + |E|)$ time. The modular decomposition of the two-structure is given by $T_1 \cap T_2 \cap ... \cap T_{k-1}$, which takes $O(k|V|)$ time to find, by Theorem 3.1.

By an only slightly more involved proof, we can get a linear time bound, as follows. Let the *essential subtree* of a partitive decomposition tree T be the tree T' obtained by deleting leaf children of the root if the root is degenerate, and let its *size* be the number of leaves in the tree.

LEMMA 3.1. *Let* $T_1, T_2, ..., T_k$ *be partitive trees on domain* V*, and let* $T'_1, T'_2, ..., T'_k$ *be their essential subtrees. It takes time proportional to the sum of sizes of* $T'_1, T'_2, ..., T'_k$ *to find* $T_1 \cap T_2 \cap ... \cap T_k$*.*

THEOREM 3.3. *It takes* $O(|V| + |E|)$ *time to find the modular decomposition of an arbitrary two-structure.*

Proof. Let $G_1, G_2, ..., G_k$ be as in the proof of Proposition 3.1. It is easy to see that the essential subtree of the modular decomposition of G_i can be obtained from the modular decomposition of the subgraph induced by non-isolated vertices, which has $O(|E_i|)$ vertices. Therefore, given E_i, the modular decomposition of G_i can be found in $O(|E_i|)$ time using the linear-time modular decomposition algorithm of [18]. Using this observation, and replacing Theorem 3.2 with Lemma 3.1 in the proof of Proposition 3.1, yields the linear time bound.

3.2 New Algebraic Operators on Symmetric Partitive Families. By Theorem 2.3, $\mathcal{N}(\mathcal{F})$ has a decomposition tree even when $\mathcal{F}$ does not have the consecutive-ones property. Therefore, $\mathcal{N}(\mathcal{F})$ is defined even when $\mathcal{F}$ is itself a symmetric partitive family.

THEOREM 3.4. *If $\mathcal{F}$ is a symmetric partitive family, then $T(\mathcal{N}(\mathcal{F}))$ is obtained from $T(\mathcal{F})$ be relabeling each degenerate node as prime and each prime node as degenerate.*

DEFINITION 3.1. *If T is the decomposition tree of a symmetric partitive family, let its* complement $\overline{T}$ *denote $T(\mathcal{N}(\mathcal{F}(T)))$. That is, $\overline{T}$ is the result of exchanging the roles of prime and degenerate nodes.*

THEOREM 3.5. *If $\mathcal{F}$ is an arbitrary set family, then $T(\mathcal{S}(\mathcal{F})) = \overline{T(\mathcal{N}(\mathcal{F}))}$.*

DEFINITION 3.2. *Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be symmetric partitive families on V, and let T_1 and T_2 be their decomposition trees. The union $\mathcal{F}_1 \cup \mathcal{F}_2$ is not necessarily partitive, so let $T_1 \cup T_2$ denote the smallest symmetric partitive family that has $\mathcal{F}_1 \cup \mathcal{F}_2$ as a subfamily, that is, let $T_1 \cup T_2 = T(\mathcal{S}(\mathcal{F}_1 \cup \mathcal{F}_2))$.*

These definitions of intersection and union therefore define a lattice on the set of all symmetric partitive trees on domain V. The minimal element of the lattice is the tree with V as its only internal node, with V labeled as prime, and the maximal element is this same tree, but with V labeled degenerate.

The following shows that the definitions satisfy familiar properties expected of these operators; the proof will appear in the journal version.

THEOREM 3.6. *Let T_1 and T_2 be decomposition trees of symmetric partitive families. Then:*

- $\overline{\overline{T_1}} = T_1$.
- $T_1 \cap T_2 = \overline{\overline{T_1} \cup \overline{T_2}}$;
- $T_1 \cup T_2 = \overline{\overline{T_1} \cap \overline{T_2}}$.

COROLLARY 3.1. *It takes $O(|V|)$ time to find the union of two symmetric partitive trees.*

Clearly, the intersection operator is commutative and associative, as is the union operator. However, together, they are not distributive. That is, it is not true in general that $T_1 \cap (T_2 \cup T_3) = (T_1 \cap T_2) \cup (T_1 \cap T_3)$, as the following example illustrates. Let $V = \{1,2,3\}$, let T_1, T_2, T_3 be decomposition trees on V where $\{1,2\}$ is the only non-root internal node of T_1, $\{1,3\}$ is the only non-root internal node of T_2, and $\{2,3\}$ is the only non-root internal node of T_3. Then $T_2 \cup T_3$ is the maximal element of the lattice, hence $T_1 \cap (T_2 \cup T_3) = T_1$. However, $T_1 \cap T_2 = T_1 \cap T_3$ is the minimal element of the lattice, hence so is $(T_1 \cap T_2) \cup (T_1 \cap T_3)$.

Let a symmetric decomposition tree on domain V be *elementary* if it has at most one non-root internal node.

THEOREM 3.7. *If T is the decomposition tree of a symmetric partitive family on domain V and T has $k \geq 1$ non-root internal nodes, then T can be written as $T = T_1\ op_1\ T_2\ op_2\ ,...,\ op_{k-1} T_k$, where each op_i is either $\cup$ or $\cap$, the operators are evaluated left-to-right, and each T_i is elementary.*

Proof. (By induction on the number of non-root internal nodes.) If there is only one non-root internal node in T, it is already elementary. Otherwise, let X be a minimal internal node, and let T' be the result of removing X from T and letting its parent, Y, adopt its children. Let T_k be the tree with root V and one other internal node, X. In T_k, V and X have the same prime/degenerate labels as Y and X, respectively, in T. If Y is degenerate, then $T = T' \cap T_k$, and if Y is prime, then $T = T' \cup T_k$. Since T' has $k-1$ non-root internal nodes, T' can be decomposed into unions and intersections of $k-1$ elementary trees to complete the expression.

Theorem 3.7 is a key element in our time bound for intersecting partitive trees (see Section 4.3).

Modules and the quotients they induce in a graph are examples of a *substitution decomposition* on the domain of graphs [22]. We can define a substitution decomposition on the domain of decomposition trees of symmetric partitive families where where the roots carry a bit that it as *strong* or *weak*. Let an *autonomous set* denote a node of T or a union of siblings in T. *Note: It is not necessary for the parent of $\mathcal{C}$ to be degenerate.* If X is autonomous, then if it is a node of T, the *factor* $T[X]$ is the subtree rooted at X, and if it is a union of a set $\mathcal{C}$ of siblings, the factor $T[X]$ is the tree where X is the root, and its subtrees are the subtrees of T rooted at members of $\mathcal{C}$; in this case, if X has at least three children, then it has the same prime/degenerate label as the parent of $\mathcal{C}$. Let the *quotient* T/X denote the operation of nodes that are subsets of X and replacing them with a single leaf. The quotient is *strong* if X is a node of T, and *weak* if it is a union of siblings that is not a node of T.

Clearly, these operations are invertible: T can be uniquely reconstructed from a quotient T/W and factor $T[W]$ if the leaf w of T/W that corresponds to W is indicated, and a bit at the root of $T[W]$ identifies whether the quotient was strong or weak.

Several algebraic properties have been described previously for substitution decompositions, but the introduction of union and intersection operators on partitive trees yields the following new identities, which we use in obtaining the new time bounds in this paper (Section 4.3):

THEOREM 3.8. *If T_a and T_b are decomposition trees of symmetric partitive families on domain V and A is autonomous in both T_a and T_b, then:*

- $(T_a \cap T_b)/A = T_a/A \cap T_b/A$.
- $(T_a \cup T_b)/A = T_a/A \cup T_b/A$.
- $(T_a \cap T_b)[A] = T_a[A] \cap T_b[A]$
- $(T_a \cup T_b)[A] = T_a[A] \cup T_b[A]$

If X and Y are disjoint autonomous sets, $(T/X)/Y = (T/Y)/X$. Therefore, we can write this as $T/\{X,Y\}$, and, more generally, if $\{A_1, A_2, ..., A_k\}$ are disjoint autonomous sets, the quotient $T/\{A_1, A_2, ..., A_k\}$ is uniquely defined.

3.3 Algorithmic Uses of Compact Representations. Any algorithm can be made to run in time linear in the size of its input simply by selecting a suitably space-inefficient representation for the input. For instance, many algorithms for NP-complete problems can be made to run in "linear" time by choosing a unary representation for integer inputs. Linearity of an algorithm does not imply an optimal time bound unless the representation of the input is also asymptotically optimal.

When Booth and Lueker's algorithm [2] for finding the PQ tree is applied to a set family that is not known to have the consecutive-ones property, the algorithm either returns the PQ tree, or else rejects the family as not having the consecutive-ones property. The running time of $O(|V| + l(\mathcal{F}))$ is an optimum time bound, since it uses a space-efficient representation of arbitrary set families.

However, when it is applied to a set family that is already known to have the consecutive-ones property, the proof of optimality of the time bound is no longer valid because it assumes an input of size $\Theta(|V| + l(\mathcal{F}))$. Families with the consecutive-ones property have a representation that is more compact than the standard listing of elements of each member of the family. A consecutive-ones family $\mathcal{F}$ can be represented in $O(|V| + |\mathcal{F}|)$ space by giving a consecutive-ones ordering, and representing each member X of $\mathcal{F}$ in $O(1)$ space by giving the first and last member of the interval occupied by X in this ordering.

THEOREM 3.9. *It takes $O(|V|+|\mathcal{F}|)$ time to find the PQ tree of a consecutive-ones family $\mathcal{F}$, given a consecutive-ones ordering and, for each $X \in \mathcal{F}$, the first and last element of X in the ordering.*

It is worth noting that Theorem 3.9 is the key starting point in the proofs of all of the remaining results of this paper. It also implies that, given the interval representation of an interval graph, the graph's PQ tree can be obtained in $O(|V|)$ time if the endpoints of the intervals are integers from 1 to $O(1)$, and in $O(|V|\log|V|)$ time if they are given as real numbers.

A similar type of result can be obtained for modular decomposition of *permutation graphs*. A permutation graph is obtained from two permutations of V, by letting the members of V be the vertices and letting two vertices x and y be adjacent if x is before y in one of the permutations and after it in the other [12]. Recognizing permutations and deriving their modular decomposition takes linear time [17]. However, it turns out that this bound for finding the modular decomposition is not optimal if the input graph is known to be a permutation graph:

THEOREM 3.10. *Given an $O(|V|)$ representation of a permutation graph using two permutations of V, it takes $O(|V|)$ time to find its modular decomposition.*

4 Sketches of proofs

4.1 Theorem 3.9. If $\mathcal{F}$ is a set family, let its *overlap graph* $G_o(\mathcal{F})$ be the graph that has one vertex for each member of $\mathcal{F}$ and an edge between two vertices iff the corresponding members of $\mathcal{F}$ overlap.

Given a connected component $\mathcal{C}$ of $G_o(\mathcal{F})$, let $\equiv_{\mathcal{C}}$ be an equivalence relation on $\bigcup\mathcal{C}$, where if $x, y \in \bigcup\mathcal{C}$, then $x \equiv_{\mathcal{C}} y$ iff the family of members of $\mathcal{C}$ that contain x is the same as the family of members of $\mathcal{C}$ that contains y. Let $\mathcal{C}$'s *blocks* be the equivalence classes of $\equiv_{\mathcal{C}}$.

THEOREM 4.1. *[16] If $\mathcal{F}$ is a set family on domain V, then $X \subseteq V$ is a node of the decomposition tree of $\mathcal{N}(\mathcal{F})$ iff it is one of the following:*

1. *V or a one-element subset of V;*
2. *$\bigcup\mathcal{C}$ for some connected component $\mathcal{C}$ of $\mathcal{F}$'s overlap graph;*
3. *A block of a connected component of $\mathcal{F}$'s overlap graph.*

By Theorem 4.1, it suffices to find the connected components of $\mathcal{F}$'s overlap graph and, for each component, find the component's union and its blocks. The sum of cardinalities of these unions and blocks is not $O(|V|+|\mathcal{F}|)$, but, since they each correspond to intervals in the consecutive-ones ordering, we can represent each of them in $O(1)$ space by giving the starting and ending position of the interval it occupies in the consecutive-ones ordering. Since the decomposition tree has $|V|$ leaves and each node of the decomposition tree of $\mathcal{N}(\mathcal{F})$ has at least two children, this takes $O(|V|)$ space.

The overlap graph does not even have $O(|V| + l(\mathcal{F}))$ size. However, it suffices to find only the components of the overlap graph. Dahlhaus has given an algorithm for finding these in $O(|V| + l(\mathcal{F}))$ time [7], but even this bound is too large for our purposes.

Each block of ones in a consecutive-ones ordering of a matrix can be viewed as an interval on the real line whose endpoints happen to be integers, namely, the column numbers of the first and last interval. It is easy to then perturb them to obtain a list of endpoints where no two endpoints coincide, without disturbing the overlap relation among the intervals.

Next, if x is an interval, let $R(x)$ denote the set of intervals that overlap with x and whose right endpoints lie to the right of x. If $R(x)$ is nonempty, let x's *right parent* be the member of $R(x)$ with the rightmost right endpoint. It's *left parent* is defined symmetrically: let $L(x)$ denote the set of intervals that overlap with x and whose left endpoints lie to the left of x. If $L(x)$ is nonempty, then x's *left parent* is the member of $L(x)$ whose left endpoint is leftmost. The *parent graph* is the graph whose vertex set is the intervals and whose edge set is $\{xy|$ one of x and y is the left or right parent of the other$\}$.

We then apply the following lemma, whose proof is omitted because of space limitations.

LEMMA 4.1. *The connected components of the parent graph are the same as the connected components of the overlap graph.*

Given the consecutive-ones arrangement and the connected components of the overlap graph, it is easy to find the blocks, hence the PQ tree, in $O(n)$ time.

4.2 Theorem 3.2.

DEFINITION 4.1. *[3] Let $\{\pi_1,\pi_2,...,\pi_k\}$ be a set of permutations of V, and let $\pi_1 = (x_1,x_2,...,x_n)$. If there exists $j < i$ such that x_p lies in between x_i and x_{i+1} in one of the permutations $\{\pi_2,...,\pi_k\}$, then let p be the minimum such j, and let us call the set $\{x_p,x_{p+1},...,x_i\}$ a* left fracture for i. *Similarly, if there exists $j' > i+1$ such that $x_{j'}$ lies between x_i and x_{i+1} in one of $\{\pi_2,...,\pi_k\}$, then let let q be the maximal such j'; $\{x_{i+1},x_{i+2},...,x_q\}$ is a* right fracture for $i+1$. *The* fractures *are the set of all left and right fractures for 1 through n.*

Clearly, the fractures have the consecutive-ones property.

LEMMA 4.2. *Let $\mathcal{F}$ be the set of fractures of a set $\{\pi_1,\pi_2,...,\pi_k\}$ of permutations of V. Then the decomposition tree of the common intervals is obtained from the PQ tree of $\mathcal{F}$ by relabeling each Q node as prime and relabeling each P node as linear. and ordering its children in the ordering in which they appear as intervals in π_1.*

The proof is omitted because of space limitations, but is not especially difficult. Lemma 4.2 and Theorem 3.9, reduce the problem of Theorem 3.2 to that of finding the fractures of $\{\pi_1,\pi_2,...,\pi_k\}$ in $O(kn)$ time. To do this, we find the fractures of $\{\pi_1,\pi_j\}$ for each j from 2 to k; the ones needed for Lemma 4.2 are a subset.

To do this, we make use of the following theorem, which is due to Gabow, Bentley, and Tarjan [10]:

THEOREM 4.2. *Given a length-n list L of real values and a set of p intervals $\{[i_1,j_1],[i_2,j_2],...,[i_p,j_p]\}$ of L, it takes $O(n+p)$ time to find a maximum element of L in each of the intervals.*

Let $\pi_1 = (x_1,x_2,...,x_n)$ and let $\pi = \pi_j = (x_{\pi(1)},x_{\pi(2)},...,y_{\pi(n)})$. Finding the right fracture of $\{\pi_1,\pi_j\}$ can be accomplished by creating a list of n integers, where the integer in position $\pi(i)$ is i. The right fracture of x_i and x_{i+1} is now just the maximum integer that occurs in the interval $[\pi(i),\pi(i+1)]$ in this list. By Theorem 4.2, we may find all right fractures of $\{\pi_1,\pi_j\}$ in $O(n)$ time, and by a symmetric operation, all left fractures in $O(n)$ time.

4.3 Theorem 3.1. Let T_a and T_b be the two trees to be intersected. To understand the approach of the general algorithm, let us first consider the case where the trees have no prime or degenerate nodes. For T_a and T_b, we may construct two permutations of V whose common intervals have T_a as their decomposition tree, as follows. Arrange each node's children according to their implied linear order. Get the first linear order by listing the elements in the leaves according to their left-to-right order in this ordered tree. Then, reverse the order of children at each node that is at odd depth in the tree and once again list the ordering of elements in the leaves to obtain the second linear order.

Similarly, we may construct two linear orders whose common intervals have T_b as their decomposition tree. It is easy to see that, together, the two permutations from T_a and the two from T_b have $T_a \cap T_b$ as the decomposition tree of their common intervals.

If we allow prime nodes, but assume that T_a and T_b have no degenerate nodes, the procedure is similar, except that three permutations may be required to represent each of T_a and T_b. We omit the details, but they are not difficult.

Let us now consider the case where no linear nodes are allowed, that is, where T_a and T_b represent symmetric partitive families.

DEFINITION 4.2. *Suppose $\mathcal{A}$ is a set of disjoint autonomous sets in a tree T, and let X be a subset of V that does not overlap any member of $\mathcal{A}$. Let R be a set consisting of one representative from each member of $\mathcal{A}$. Then the set $X' = X \cap ((V - \bigcup \mathcal{A}) \cup R)$ is the* homomorphic image *of X in the domain of $T_a/\mathcal{A}$ and $T_b/\mathcal{A}$.*

To find $T_a \cap T_b$, let us write T_a as a composition $T_1\ op_1\ T_2\ op_2,\ ...,\ op_{k-1}\ T_k$ of elementary decomposition trees, using Theorem 3.7. As in the constructive

proof of the theorem, we may assume that the non-root internal node of T_k is a minimal internal node X of T_a, that is, a node that has only leaf children. Let $T'_a = T_1\ op_1\ T_2\ op_2$, ..., $op_{k-2}\ T_{k-1}$, so that T_a can be written as $T'_a\ op_{k-1}\ T_k$. It is easy to see from the proof of Theorem 3.7 that T'_a is obtained from T_a by removing X and letting its parent inherit its children.

Let T'_k be the result of changing the root of T_k to degenerate, if it is not already degenerate. Let $\mathcal{A}$ be the maximal unions of siblings of T_b that are subsets of X. Then each member of $\mathcal{A}$ is autonomous T_a, T'_a, T'_k, and T_b. Let X' and V' be the homomorphic images of X and V, respectively, under the quotient by $\mathcal{A}$. X' can be represented by selecting one representative vertex from each member of $\mathcal{A}$, and V' can be represented as $(V-X)\cup X'$. Note that X' is a node of $T_a/\mathcal{A}$, and V' is the domain of $T_a/\mathcal{A}$, $T'_a/\mathcal{A}$, $T'_k/\mathcal{A}$, and $T_b/\mathcal{A}$.

The following is not hard to prove using the algebraic operators of Section 3.2:

LEMMA 4.3. $(T_a \cap T_b)/\mathcal{A} = T'_a/\mathcal{A} \cap T'_k/\mathcal{A} \cap T_b/\mathcal{A}$.

ALGORITHM 4.1. Find the intersection $T_a \cap T_b$ of two decomposition trees of symmetric partitive families.

1. Choose a minimal internal node X of T_a.
2. Find the family $\mathcal{A}$ of maximal autonomous sets of T_b that are subsets of $\mathcal{A}$, let T'_a and T'_k be as in Lemma 4.3.
3. Using the simple structure of T'_k, find $T'_b = T'_k/\mathcal{A} \cap T_b/\mathcal{A}$.
4. Find $(T_a \cap T_b)/\mathcal{A}$ by recursively evaluating $T'_a/\mathcal{A} \cap T'_b$.
5. Find $(T_a \cap T_b)[Y]$ for each $Y \in \mathcal{A}$.
6. Substitute these trees into $(T_a \cap T_b)/\mathcal{A}$ to obtain $T_a \cap T_b$.

Steps 1 and 6 are obvious. Step 5 is trivial: $(T_a \cap T_b)[Y]$ is equal to $T_a[Y]$ if Y is prime, and it is equal to $T_b[Y]$ if Y is degenerate. Step 4 is solved by recursion, and works by induction on the number of nodes of T_a; its correctness follows from Lemma 4.3. The difficulty for the time bound is Steps 2 and 3.

It is easy to get an $O(|X|)$ bound for Step 2 using the following trick, which is given in a variety of sources, such as [18]:

LEMMA 4.4. *Given $X \subseteq V$ and the inclusion tree of a tree-like family $\mathcal{F}$, it takes $O(|X|)$ time to find the maximal members of $\mathcal{F}$ that are subsets of X.*

To obtain the $O(|V|)$ time bound for Algorithm 4.1, we use an *amortized analysis* [24, 5]. Let $n(T)$ denote the number of nodes of of a decomposition tree T, let $\mathcal{D}(T)$ denote the set of degenerate nodes, and if $Z \in \mathcal{D}(T)$ let $d(Z,T)$ denote the number of children of Z in T. Let us define a potential function $\phi(T_b) = n(T_b) + \sum_{Z\in\mathcal{D}(T_b)}(d(Z,T_b)-1)$. Initially, this is $O(|V|)$ and it is positive for every tree passed as T_b to a recursive call.

Let $\mathcal{N}$ be the set of nodes that are both proper ancestors of leaf subsets of X' in $T_b/\mathcal{A}$ and proper descendants of their least common ancestor. Since no internal node has more than one leaf child in X', it is easy to see that $|\mathcal{N}| = \Omega(|X'|)$. Using Lemma 4.4, we must spend $O(|X|)$ time finding $\mathcal{A}$. In addition, we show that we must spend $O(1)$ time operating on each each member of $\mathcal{N}$. However, we also show that each prime node in $\mathcal{N}$ is destroyed, reducing the potential by 1 ($n(T_b/\mathcal{A})$ decreases), and each degenerate node either loses a child or is split into two degenerate nodes ($\sum_{Z\in\mathcal{D}(T_b/\mathcal{A})}(d(Z,T_b/\mathcal{A})-1)$ decreases). This drop in potential covers the cost of operating on members of $\mathcal{N}$, and $\Omega(|X'|)$ of the $O(|X|)$ cost of applying Lemma 4.4. The remaining $O(|X|-|X'|)$ cost is covered by the decrease of $\Omega(|X|-|X'|)$ in going from $\phi(T_b)$ to $\phi(T_b/\mathcal{A})$, which reduces in the number of nodes by that amount.

This gives $O(n)$ algorithms for finding the intersection of symmetric partitive families when the nodes are prime and linear, or prime and degenerate. The case where T_a and T_b can have all three types of nodes can be solved in $O(n)$ time by a combination of these steps; full details will be given in the journal version.

5 Conclusions and Open Problems

Though they have allowed us to get improved bounds for problems involving arbitrary partitive families, the operators of Section 3.2 have been defined only for symmetric partitive families. An obvious question is how to to generalize them to arbitrary partitive families.

A definition of the intersection of two partitive is immediate from Theorem 2.4. One way to define the union of arbitrary partitive trees T_1 and T_2 is $T_1 \cup T_2 = \mathcal{C}(\mathcal{F}(T_1) \cup \mathcal{F}(T_2))$.

Here is one argument in favor of this definition. In Application 3.2, the algorithm returns the decomposition tree of a symmetric partitive tree as the PQ tree, and therefore fails to assign the orderings to children of Q nodes. This can be added in a separate step. However, a simpler alternative is to compute the PQR tree for the top half M_1 of the matrix and for the bottom half M_2, interpret the P nodes as prime, the Q nodes as linear, and R nodes as degenerate, and compute the union of these two trees as we have just defined it. It is immediate from the definition of the PQR tree that this is the PQR tree of the whole matrix. If the matrix has the consecutive-ones property, there are no R nodes, and the linear order on children of linear nodes gives the desired ordering of children of Q nodes.

In addition, we claim that, under these definitions of

union and intersection, Theorem 3.7 generalizes easily to arbitrary partitive families.

One disadvantage of this definition of the union is that it is not a generalization of the definition of Section 3.2, since the union of two symmetric partitive families need not be symmetric: if $V = \{a, b, c\}$, T_1 has $\{a, b\}$ as an internal node, and T_2 has $\{b, c\}$ as an internal node, then the union has a linear root. Thus, this union operation gives a different result from that of Section 3.2. Also, we do not know of a reasonable definition of the complement operation that gives an analog of Theorem 3.6 on arbitrary partitive families. If this is possible, it may require not just a definition of the complement, but also a different definition of the union than the one we have suggested here.

References

[1] S. Benzer. On the topology of the genetic fine structure. *Proc. Nat. Acad. Sci. U.S.A.*, 45:1607–1620, 1959.

[2] S. Booth and S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, 13:335–379, 1976.

[3] C. Capelle, M. Habib, and F. d Montgolfier. Graph decomposition and factorising permutations. *Discrete Mathematics and Theoretical Computer Science*, 5:55–70, 2002.

[4] M. Chein, M. Habib, and M. C. Maurer. Partitive hypergraphs. *Discrete Mathematics*, 37:35–50, 1981.

[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw Hill, Boston, 2001.

[6] M. Crochemore and W. Rytter. *Text Algorithms*. Oxford University Press, Oxfort, 1994.

[7] E. Dahlhaus. Parallel algorithms for hierarchical clustering, and applications to split decomposition and parity graph recognition. *Journal of Algorithms*, 36:205–240, 2000.

[8] A. Ehrenfeucht, H. N. Gabow, R. M. McConnell, and S. J. Sullivan. An $O(n^2)$ divide-and-conquer algorithm for the prime tree decomposition of two-structures and modular decomposition of graphs. *Journal of Algorithms*, 16:283–294, 1994.

[9] A. Ehrenfeucht and G. Rozenberg. Theory of 2-structures, part 2: Representations through labeled tree families. *Theoretical Computer Science*, 70:305–342, 1990.

[10] H.N. Gabow, J.L. Bentley, and R. E. Tarjan. Scaling and related techniques for geometry problems. *Proceedings of the ACM symposium on theory of computing*, 16:135–143, 1984.

[11] T. Gallai. Transitiv orientierbare Graphen. *Acta Math. Acad. Sci. Hungar.*, 18:25–66, 1967.

[12] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

[13] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, Cambridge, 1997.

[14] W.L. Hsu and R.M. McConnell. PC trees and circular-ones arrangements. *Theoretical Computer Science*, 296:59–74, 2003.

[15] R. M. McConnell. Linear-time recognition of circular-arc graphs. *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS01)*, 42:386–394, 2001.

[16] R. M. McConnell. A certifying algorithm for the consecutive-ones property. *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA04)*, 15:to appear, 2004.

[17] R. M. McConnell and J. P. Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201(1-3):189–241, 1999.

[18] R.M. McConnell and F de Montgolfier. Linear-time modular decomposition of directed graphs. *Discrete Applied Math, to appear*.

[19] R.M. McConnell and J.P. Spinrad. Construction of probe interval models. *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 866–875, 2002.

[20] F.R. McMorris, C. Wang, and P. Zhang. On probe interval graphs. *Discrete Applied Mathematics*, 88:315–324, 1998.

[21] R. H. Möhring. Algorithmic aspects of comparability graphs and interval graphs. In I. Rival, editor, *Graphs and Order*, pages 41–101. D. Reidel, Boston, 1985.

[22] R. H. Möhring. Algorithmic aspects of the substitution decomposition in optimization over relations, set systems and boolean functions. *Annals of Operations Research*, 4:195–225, 1985.

[23] R. H. Möhring and F. J. Radermacher. Substitution decomposition for discrete structures and connections with combinatorial optimization. *Annals of Discrete Mathematics*, 19:257–356, 1984.

[24] R. E. Tarjan. *Data structures and network algorithms*. Society for Industrial and Applied Math., Philadelphia, 1983.

[25] P. Zhang. United states patent: Method of mapping DNA fragments, available at www.cc.columbia.edu/cu/cie/techlists/patents/5667970.htm. July 3, 2000.